<https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-2-populating-a-table-with-existing-hierarchical-data>

# Lesson 1-2 - Populating a Table with Existing Hierarchical Data

2017-3-6 2 min to read Contributors

* [BYHAM](https://github.com/BYHAM)

* [Craig Guyer](https://github.com/craigg-msft)

This task creates a new table and populates it with the data in the **EmployeeDemo** table. This task has the following steps:

* Create a new table that contains a **hierarchyid** column. This column could replace the existing **EmployeeID** and **ManagerID** columns. However, you will retain those columns. This is because existing applications might refer to those columns, and also to help you understand the data after the transfer. The table definition specifies that **OrgNode** is the primary key, which requires the column to contain unique values. The clustered index on the **OrgNode** column will store the date in **OrgNode** sequence.
* Create a temporary table that is used to track how many employees report directly to each manager.
* Populate the new table by using data from the **EmployeeDemo** table.

### To create a new table named NewOrg

* In a Query Editor window, run the following code to create a new table named **HumanResources.NewOrg**:

Copy

CREATE TABLE NewOrg

(

OrgNode hierarchyid,

EmployeeID int,

LoginID nvarchar(50),

ManagerID int

CONSTRAINT PK\_NewOrg\_OrgNode

PRIMARY KEY CLUSTERED (OrgNode)

);

GO

### To create a temporary table named #Children

1. Create a temporary table named **#Children** with a column named **Num** that will contain the number of children for each node:

Copy

CREATE TABLE #Children

(

EmployeeID int,

ManagerID int,

Num int

);

GO

1. Add an index that will significantly speed up the query that populates the **NewOrg** table:

Copy

CREATE CLUSTERED INDEX tmpind ON #Children(ManagerID, EmployeeID);

GO

### To populate the NewOrg table

1. Recursive queries forbid subqueries with aggregates. Instead, populate the **#Children** table with the following code, which uses the [ROW\_NUMBER()](https://docs.microsoft.com/en-us/sql/t-sql/functions/row-number-transact-sql) method to populate the **Num** column:

Copy

INSERT #Children (EmployeeID, ManagerID, Num)

SELECT EmployeeID, ManagerID,

ROW\_NUMBER() OVER (PARTITION BY ManagerID ORDER BY ManagerID)

FROM EmployeeDemo

GO

1. Review the **#Children** table. Note how the **Num** column contains sequential numbers for each manager.

Copy

SELECT \* FROM #Children ORDER BY ManagerID, Num

GO

Here is the result set.

EmployeeID ManagerID Num

---------- --------- ---

1 NULL 1

2 1 1

3 1 2

4 2 1

5 2 2

6 2 3

7 3 1

8 3 2

9 4 1

10 4 2

1. Populate the **NewOrg** table. Use the GetRoot and ToString methods to concatenate the **Num** values into the **hierarchyid** format, and then update the **OrgNode** column with the resultant hierarchical values:

Copy

WITH paths(path, EmployeeID)

AS (

-- This section provides the value for the root of the hierarchy

SELECT hierarchyid::GetRoot() AS OrgNode, EmployeeID

FROM #Children AS C

WHERE ManagerID IS NULL

UNION ALL

-- This section provides values for all nodes except the root

SELECT

CAST(p.path.ToString() + CAST(C.Num AS varchar(30)) + '/' AS hierarchyid),

C.EmployeeID

FROM #Children AS C

JOIN paths AS p

ON C.ManagerID = P.EmployeeID

)

INSERT NewOrg (OrgNode, O.EmployeeID, O.LoginID, O.ManagerID)

SELECT P.path, O.EmployeeID, O.LoginID, O.ManagerID

FROM EmployeeDemo AS O

JOIN Paths AS P

ON O.EmployeeID = P.EmployeeID

GO

1. A **hierarchyid** column is more understandable when you convert it to character format. Review the data in the **NewOrg** table by executing the following code, which contains two representations of the **OrgNode** column:

Copy

SELECT OrgNode.ToString() AS LogicalNode, \*

FROM NewOrg

ORDER BY LogicalNode;

GO

The **LogicalNode** column converts the **hierarchyid** column into a more readable text form that represents the hierarchy. In the remaining tasks, you will use the ToString() method to show the logical format of the **hierarchyid** columns.

1. Drop the temporary table, which is no longer needed:

Copy

DROP TABLE #Children

GO

The next task will create indexes to support the hierarchical structure.

## Next Task in Lesson

[Optimizing the NewOrg Table](https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-3-optimizing-the-neworg-table)